IN THE CLAIMS

52. (Currently Amended) A method for preparing executables for execution in a distributed processing environment in accordance with a set of process representations of business logic, the method comprising the steps of:

providing process representations in a process calculus notation, the process representations specifying one or more environmental constraints;

verifying that the representations are valid;

generating executables and corresponding test data in accordance with the verified representations;

testing the executables using the corresponding test data;

deploying the tested executables in the distributed processing environment;

monitoring the performance of the deployed executables to gather process execution information;

analysing information gathered in the monitoring step; and

autonomically altering the executables and corresponding test data in accordance with analysed process execution information and contextual information.

53. (Previously Presented) A method according to claim 52, wherein the step of autonomically altering the executables includes altering the generation of executables and test data in accordance with analysed process execution information.

54. (Previously Presented) A method according to claim 52, wherein the step of autonomically altering the executables comprises altering the process representations and repeating the verification, generation and testing steps.

55. (Previously Presented) A method according to claim 52, wherein the step of autonomically altering the executables is furthermore performed in accordance with contextual information.

56. (Previously Presented) A method according to claim 55, wherein the contextual information includes heuristics.

57. (Previously Presented) A method according to claim 52, wherein the step of generating the executables is furthermore performed in accordance with contextual information.

58. (Previously Presented) A method according to claim 52, wherein the step of autonomically altering the executables comprises altering the executables and test data directly and repeating the testing step.

59. (Previously Presented) A method according to claim 52, wherein the step of autonomically altering the executables comprises comparing a set of representations of the analysed
process execution information with an earlier set of process representations and altering the executables to reduce significant disparities between them.

60. (Previously Presented) A method according to claim 59, further comprising the step of repeating the generating, testing, analysing and altering steps until the comparison indicates the absence of significant disparity.

61. (Previously Presented) A method according to claim 52, wherein the step of generating the executables comprises: generating an intermediate version of the verified representation in a third generation language; and compiling the intermediate version into the executables.

62. (Previously Presented) A method according to claim 61, wherein the third generation language is one of the set of languages including C, C++, C#, Ada, Java, Delphi, Visual Basic, and FORTRAN 90.

63. (Previously Presented) A method according to claim 52, wherein the step of generating the executables comprises generating the executables directly.

64. (Previously Presented) A method according to claim 52, wherein the process calculus notation is based upon XML.

65. (Currently amended) A computer program product for preparing executables for execution in a distributed processing environment in accordance with a set of process representations of business logic, the product comprising:

a datastore for storing process representations in a process calculus notation, the process representations specifying one or more environmental constraints;

a verification module, for verifying that the representations are valid;

a generator module for generating executables and corresponding test data in accordance with the verified representations;

a tester module for testing the executables using the corresponding test data, wherein the tester module allows the tested executables to be deployed in the distributed processing environment;

a monitor module, for monitoring the performance of the deployed executables to gather process execution information;

an analyser module for analysing process execution information gathered by the monitor module; and

an update module, which alters the executables and corresponding test data autonomically in accordance with analysed process execution information and contextual information.

66. (Previously Presented) A computer program product according to claim 65, wherein the autonomic alteration of the executables and corresponding test data includes alteration of the behaviour of the generator module in accordance with analysed process execution information.

67. (Previously Presented) A computer program product according to claims 65, wherein the update module autonomically alters the executables and corresponding test data by altering the process representations and instigating the sequential operation of the verification module, the generator module, the tester module and the analyser module.

68. (Previously Presented) A computer program product according to claim 65, wherein the update module autonomically alters the executables in accordance with contextual information.

69. (Previously Presented) A computer program product according to claim 68, wherein the contextual information includes heuristics.

70. (Previously Presented) A computer program product according to claim 65, wherein the generator module generates the executables in accordance with contextual information.

71. (Previously Presented) A computer program product according to claim 65, wherein the update module autonomically alters the executables and corresponding test data by altering the executables and test data directly and instigating the operation of the tester module and the analyser module.

72. (Previously Presented) A computer program product according to claim 65, wherein the update module autonomically alters the executables and corresponding test data by comparing a set of representations of the analysed process execution information with the earlier set of process representations and altering the executables to reduce significant disparities between them.

73. (Previously Presented) A computer program product according to claim 72, wherein the sequential operation of the verification module, the generator module, the tester module, the analyser module and the update module continues autonomically until the comparison indicates the absence of significant disparity.

74. (Previously Presented) A computer program product according to claim 65, wherein the generator module generates the executables by generating an intermediate version of the verified representation in a third generation language and then compiling the intermediate version into the executables.

75. (Previously Presented) A computer program product according to claim 74, wherein the third generation language is one of the set of languages including C, C++, C#, Ada, Java, Delphi, Visual Basic, and FORTRAN 90.

76. (Previously Presented) A computer program product according to claim 65, wherein the generator module generates the executables directly.

77. (Previously Presented) A computer program product according to claim 65, wherein the process calculus notation is based upon XML.

78. (Currently Amended) A computer system for preparing executables for execution in a distributed processing environment in accordance with a set of process representations of business logic, the system comprising:

a datastore for storing process representations in a process calculus notation, the process representations specifying one or more environmental constraints;

a verification module, for verifying that the representations are valid;

a generator module for generating executables and corresponding test data in accordance with the verified representations;

a tester module for testing the executables using the corresponding test data, wherein the tester module allows the tested executables to be deployed in the distributed processing environment;

a monitor module, for monitoring the performance of the deployed executables to gather process execution information;

an analyser module for analysing process execution information gathered by the monitor module; and

an update module, which alters the executables and corresponding test data autonomically in accordance with analysed process execution information and contextual information.

79. (Previously Presented) A computer system according to claim 78, wherein the autonomic alteration of the executables and corresponding test data includes alteration of the behaviour of the generator module in accordance with analysed process execution information.

80. (Previously Presented) A computer system according to claim 78, wherein the update module autonomically alters the executables and corresponding test data by altering the process representations and instigating the sequential operation of the verification module, the generator module, the tester module and the analyser module.

81. (Previously Presented) A computer system according to claims 78, wherein the update module autonomically alters the executables in accordance with contextual information.

82. (Previously Presented) A computer system according to claim 81, wherein the contextual information includes heuristics.

83. (Previously Presented) A computer system according to claim 78, wherein the generator module generates the executables in accordance with contextual information.

84. (Previously Presented) A computer system according to claim 78, wherein the update module autonomically alters the executables and corresponding test data by altering the executables and test data directly and instigating the operation of the tester module and the analyser module.

85. (Previously Presented) A computer system according to claim 78, wherein the update module autonomically alters the executables and corresponding test data by comparing a set of representations of the analysed process execution information with the earlier set of process representations and altering the executables to reduce significant disparities between them.

86. (Previously Presented) A computer system according to claim 85, wherein the sequential operation of the verification module, the generator module, the tester module, the analyser module and the update module continues autonomically until the comparison indicates the absence of significant disparity.

87. (Previously Presented) A computer system according to claim 78, wherein the generator module generates the executables by generating an intermediate version of the verified representation in a third generation language and then compiling the intermediate version into the executables.

88. (Previously Presented) A computer system according to claim 87, wherein the third generation language is one of the set of languages including C, C++, C#, Ada, Java, Delphi, Visual Basic, and FORTRAN 90.

89. (Previously Presented) A computer system according to claim 78, wherein the generator module generates the executables directly.

90. (Previously Presented) A computer system according to claim 78, wherein the process calculus notation is based upon XML.